1 Forelesning 4

1.1 Rekursjon - Fibonacci

1.2 Binære trær

Et binært tres antall noder dobles for hvert nivå man går nedover treet.

1.3 Binærsøk

Problem: Sortert tabell med n element, leter etter et bestemt element. Hvor mange ganger må vi teste før vi er sikker på å finne det?

1. Forsøk - n element igjen: prøver midterste element
2. Forsøk - $\frac{n}{2}$ element igjen: prøver igjen midterste element

Osv, til det kun er ett element igjen.

1.4 Sortering - hvorfor?

1.4.1 Hva er sortering?

Sortering er en total ordning av en mengde A. F. eks følgende ordning

$$a_1 \leq a_2 \leq a_3 \leq \cdots \leq a_n$$

Ordningen kan være anti-symmetrisk.

$$-a \leq b \land a \geq b \Rightarrow a = b$$

Eller Transitiv

$$a \leq b \land b \leq c \rightarrow a \leq c$$

Eller Total

$$a \leq b \lor b \leq a$$

1.5 Sortering i Java

Man kan sortere med f. eks `Collections.sort(List <T> list)`, men hvilken type må `T` være? Man må implementere Comparable. Med Java generics blir det litt mer komplisert, men man kan implementere comparable på følgende måte.

```
T extends Comparable <?  super T>
```

1.5.1 Comparable Interface

- `public int compareTo(T obj);`
  - Returnerer -1 hvis `this < obj`
  - Returnerer 0 hvis `this == obj`
  - Returnerer 1 hvis `this > obj`
- Beskriver naturlig ordning
  - Integer, Double, String, Date...

1.5.2 Sortere på forskjellige måter

Da må man bruke `Comparator interface`

```
int compare(T o1, T o2)
```

1.5.3 Mange sorteringsalgoritmer

- Selection sort
- Merge Sort
- Heap Sort
- Bogo Sort (The GOAT)
- Insertion Sort
- Shell Sort
- Quick Sort
- Bucket Sort
- Radix Sort
- Bubble Sort